12-05-05

AF

PTO/SB/21 (09-04)
Approved for use through 07/31/2006. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission: 27

Application Number	09/928,881
Filing Date	08/13/2001
First Named Inventor	Konstantin Konston
Art Unit	2186
Examiner Name	Tsai, Sheng Jen
Attorney Docket Number	DE920000074US1

ENCLOSURES *(Check all that apply)*

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment/Reply
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Reply to Missing Parts/ Incomplete Application
  - ☐ Reply to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
  - ☐ Landscape Table on CD
- ☐ After Allowance Communication to TC
- ☑ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☑ Other Enclosure(s) (please Identify below):
  POST CARD

Remarks

Please charge the appeal fee of $500.00 and any other fees due or credit any overpayment made ti deposit account 50-0510.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	Holland & Knight, LLP		
Signature	Michael J. Buchenhorner		
Printed name	Michael J. Buchenhorner		
Date	November 22, 2005	Reg. No.	33,162

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature	Michael J. Buchenhorner		
Typed or printed name	Michael J. Buchenhorner	Date	November 22, 2005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT(S):	Konstatin Konson
SERIAL NUMBER:	09/928,881
FILING DATE:	August 13, 2001
TITLE:	Method and System for Persistently Storing Objects in an Object Oriented Environment
GROUP ART UNIT:	2186
TECHNOLOGY CENTER:	2100
ATTORNEY DOCKET NO.:	DE920000074US1

APPEAL BRIEF

CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

November 22, 2005
Date of Deposit

Michael J. Buchenhorner
Signature

Michael J. Buchenhorner
Typed or Printed Name of Person Signing Certificate

TABLE OF CONTENTS

1. REAL PARTY IN INTEREST

The real party in interest is Appellant, International Business Machines Corporation.

2. RELATED APPEALS AND INTERFERENCES

Upon information and belief, there are no other appeals or interferences that will directly affect or be directly affected by or having a bearing on the Board's decision in this pending Appeal.

3. STATUS OF THE CLAIMS

Claims 1-24 are pending. Claim 1 was objected to for certain alleged informalities. All claims 1-24 stand rejected as anticipated and have been appealed.

4. STATUS OF AMENDMENTS

No substantive amendments have been made since the final office action dated May 12, 2005.

5. SUMMARY OF CLAIMED SUBJECT MATTER

The subject invention is a method and system for persistently storing and restoring persistently stored objects (see page 4, lines 22-25). In a first embodiment set forth in claim 1, a method for restoring persistently stored objects of an object-oriented environment established in a computer system having a volatile memory

and a persistent storage, the method comprises the steps of: (a) retrieving from said persistent storage a first list (page 5, lines 23-24) comprising first references to segments and stored in said persistent storage (page 5, lines 24-25); (b) retrieving all segments referenced by said first references and storing them in said volatile memory (page 5, lines 23-24); (c) saving in the first list the difference between an old memory address, at which the segment used to reside in the volatile memory, and a new memory address at which the segment is stored (page 5, lines 25-29); (d) retrieving from the persistent storage a second list comprising second references to blocks, whereby the blocks contain an object description (page 5, lines 30-31); (e) determining which segment contains the block referenced by a particular element of the second list (page 6, lines 1-3); (f) creating an object in said volatile memory using said object description from said segment and saving a new address of said created object in said second list in volatile memory (page 6, lines 3-5); (g) initializing said new object with values taken from the object description (page 6, lines 6-7); and (h) determining the new addresses of the new object referenced by the newly-created object and setting the new address as the reference in said new object (page 6, lines 8-9).

In a second embodiment the invention set forth in claim 14, a method for persistently storing objects of an object-oriented environment established on a computer system having a volatile memory and a persistent storage, the method comprises steps of: (a) allocating in the volatile memory, segments (page 4, lines 26-27, Fig. 4 element 320); (b) creating a first list comprising first references to the segments (page 4, lines 27-29); (c) creating a second list (page 5, lines 3-4) comprising second references to blocks (page 5 line 2 and e.g., blocks 324 and 325

in Fig. 4) wherein the blocks are portions of said segments (Fig. 4, segment 320 blocks 322-325); (d) allocating a block of one of said segments (page 5, line 6); (e) creating an object description (page 5, line 6, e.g., object descriptions 221-228 in Fig. 3) for an object by saving values owned by the object of the variables belonging to its class into said allocated block (page 5, lines 7-8); (f) adding a new element to said second list containing the particular reference to said object description (page 5, lines 9-10); determining the address of another object description of another object referenced in said object (page 5, lines 11-13); (g) setting the address of said respective object description as the reference in the created object description (page 5, line 15-16); (h) storing said second list on said persistent storage (page 5, lines 18-19); (i) storing the segments referenced by the first list on the persistent storage; and storing the first list on said persistent storage (page 5, lines 23-25).

6. GROUNDS OF REJECTION TO BE REVIEWED

I. Whether the Examiner erred in objecting to claim 1 under 35 USC §112, as informal for confusing the claimed blocks and segments.

!!. Whether the Examiner erred in rejecting claims 1-24 under 35 U.S.C. §102 as being anticipated by U.S. Patent No. 5,752,243 (Reiter).

7. ARGUMENT

I. The Examiner erred in objecting to claim 1 under 35 USC §112, as informal for confusing the claimed blocks and segments.

The Examiner objected to claim 1 as informal. Specifically, the Examiner contends that in claim 1 there is confusion between the terms "block" and "segment."

Appellant respectfully disagrees and notes that the terms are defined in the specification and hence there is no need to unnecessarily clutter the claims with the definition of those terms. The written description states that "Each segment is subdivided into a plurality of portions, so called blocks." See specification at page 9, lines 16-17. Therefore, segments contain blocks. Moreover, blocks contain a representation of an object. See specification at page 10, line 12. The Examiner correctly notes in the final office action that the applicant did not intend the terms "segment" and "block" to be the same. However, the perceived ambiguity disappears when the above is understood and any claim construction interpreting "segment" and "block" to be the same would be an improper claim construction because it would violate the canon of claim construction that rejects any claim construction that does not read on the written description. *See Vitronics Corp. v. Conceptronic, Inc.* 90 F.2d 1576 (Fed. Cir. 1996). However, the Examiner's stated claim construction: "the examiner will interpret that 'object description' is contained in a block instead of a segment ..." is incorrect because it violates the above canon. In fact, an object description is contained in one or more blocks, which are in turn contained in a segment. See specification at page 10, lines 14-16. Therefore, the objection to claim 1 should be reversed. In this reversal will also require reversal of the other rejections because an improper claim construction, i.e., an improper determination of the scope of the claims, can distort an entire analysis. Moeller v. Ionetics, Inc., 794 F.2d 653, 656, 229 USPQ 992, 994 (Fed. Cir. 1986).

II. The Examiner Erred in Rejecting Claims 1-24 under 35 U.S.C. §102 as anticipated by the Reiter Patent.

The Final Office Action has rejected claims 1-24 under 35 U.S.C. §102 as being anticipated by U.S. Patent 5,752,243 issued to Reiter, et al (hereafter, "Reiter"). Applicant respectfully requests reversal of the rejection for the following reasons.

Anticipation requires that each and every claim limitation be present in a single piece of prior art. *In re Paulsen*, 30 F.3d 1475 (Fed. Cir. 1990). The Examiner has not shown that Reiter either teaches or suggests a method as set forth in claim 1. Reiter relates to a method and structure for accessing multidimensional data in secondary storage (col. 2, line 66 – col. 3, line 3). Reiter neither teaches nor suggests the invention as claimed.

The Examiner has not shown that "retrieving from said persistent storage a first list comprising first references to segments, stored in said persistent storage" reads on the discussion at col. 6, lines 56-64 of Reiter. The first list contains references to segments, which in turn contain blocks. The Examiner does not show how this step is performed by Reiter. First, Reiter does not disclose a "first list" at all. The Examiner appears to contend that either a page from a MDB-tree or a "key value table" corresponds to "first list." The "key value table" is shown in figures 3A-D and discussed at col. 4, lines 55-57 of Reiter. The "first list" contains references to "segments" but the Examiner has not shown that Reiter teaches or suggests any "segments." By contrast, the key table of Reiter holds two key values used to perform a search of a tree. The Examiner apparently contends that segments are stored in data area 28 of figure 3A. Final Office Action at page 5. However, Reiter

states that data area 28 "holds data for which it is desirable to directly incorporate the data into the node." Col. 4, lines 60-62. That neither teaches nor suggests that segments as claimed are stored therein. Claim 1 requires retrieving the first list from persistent storage. The Examiner has not shown that the key value table (the alleged first list) is stored in persistent storage. The Examiner refers to figure 2 which shows a persistent storage but does not suggest that the key value table resides therein.

The Examiner has not shown the step of "retrieving all segments referenced by said first references and storing them in said volatile memory." The Examiner has not identified anything in Reiter that teaches or suggests this element. As established above, Reiter neither teaches nor suggests "segments." The Examiner apparently contends that the data area 28 (Fig. 3A) is an area for storing segments. However, the key table of Reiter does not point to the data area 28 as it would have to if it were the "first list." Instead, Fig. 3B shows that key values relate to key terms in other nodes. Thus, the Examiner erred in contending that the second step of claim 1 reads on Reiter.

The Examiner cites col. 3, lines 3-6 and that has nothing to do with storing anything in volatile memory. The Examiner improperly overlooked the requirement of "storing in volatile memory" and hence erroneously read this claim element on Reiter.

The third step of claim1 requires "saving in said first list the difference between an old memory address, at which the segment used to reside in the volatile memory, and a new memory address at which said segment is stored." The Examiner has not even attempted to show how Reiter teaches or suggests this

element. This is not surprising because Reiter says nothing about an old memory address or a new memory address. Instead the Examiner merely repeated the claim language at page 5 of the final Examiner. This failure requires reversal.

The fourth step of claim 1 requires "retrieving from said persistent storage a second list comprising second references to blocks, whereby said blocks contain an object description." Again, the Examiner has failed to show how Reiter teaches or suggests this element. The Examiner apparently contends that the subnode table 27 of Figs. 3A-D corresponds to the second list. The "second list" refers to blocks containing an object description. However, the Examiner has not shown teaches or suggests this. Rather, Reiter is read from storage in page units. Col. 6, lines 63-64. A page unit can include more than one node (see Figs. 3d and 5) and hence can include more than one The Examiner has not explained how to distinguish among various key tables and subnode tables as compliance with claim 1 would require.

The Examiner further failed to show how Reiter teaches or suggests the claim step of "determining which segment contains said block referenced by a particular element of said second list." The Examiner cites Fig. 5 of Reiter but failed to show how any determination of a segment is made in Reiter. Instead the Examiner concluded that Fig. 5 shows objects pointing at each other. Fig. 5, actually contradicts the Examiner's contention by showing multiple nodes in some pages that are downloaded by Reiter, as noted above.

The Examiner further failed to show how Reiter teaches or suggests the claim step of "creating an object in said volatile memory using said object description from said segment and saving a new address of said created object in said second list in

volatile memory." The Examiner cites Fig. 8B, however, no step of Fig. 8B, even mentions creating a new object, let alone doing so using the object description.

The Examiner further failed to show how Reiter teaches or suggests the claim step of "initializing said new object with values taken from said object description." The Examiner cites Fig. 8B, step 172. However, the discussion of that step states that step 172 selects a weight greater than 1kb and that has nothing to do with initialization of an object.

The Examiner further failed to show how Reiter teaches or suggests the claim step of "determining said new addresses of said new object referenced by the newly-created object and setting said new address as the reference in said new object." The Examiner cited Fig. 8B, step 176. However step 176 stores a pointer in the subnode table of a selected subtree's parent node. The Examiner contends that a pointer node is a new object. However, that misconstrues the claim because it is notoriously well known that a pointer is not an object. Appellant's application defines an object as a "unique instance of a data structure defined according to a template provided by its class." Specification at page 1. A pointer node is not an instance of any data structure. See Reiter Fig. 3C, item 38 (a pointer node holds no data). Therefore, this rejection should be reversed.

Claims 2-13 depend on claim 1 and are not anticipated by Reiter at least for the reasons discussed above.

Claim 14, as amended, relates to a method for persistently storing objects of an object-oriented environment established on a computer system having a volatile memory and a persistent storage. Reiter neither teaches nor suggests the invention of claim 14 and therefore, the rejection should be reversed.

The Examiner failed to show how Reiter teaches or suggests the step of claim 14 of "allocating in said volatile memory, segments" The Examiner cites Fig. 4 in support. However, as established above, Reiter neither teaches nor suggests "segments." Moreover, the discussion of Reiter cites states that the MDB tree is written to secondary memory, which is persistent storage, not volatile memory as required.

The Examiner further failed to show how Reiter teaches or suggests the claim step of "creating a first list comprising first references to said segments." The Examiner cites Fig. 4 in support. As established above, the key value table does not correspond to the claimed "first list."

The Examiner further failed to show how Reiter teaches or suggests the claim step of "creating a second list comprising second references to blocks." Again, the Examiner cites Fig. 4 in support. As established above, the subnode table does not correspond to the claimed "second list." Moreover, the subnodes of Reiter do not correspond to the claimed blocks, which hold object descriptions and the Examiner has not shown how Reiter holds object descriptions.

The Examiner further failed to show how Reiter teaches or suggests the claim step of "allocating a block of one of said segments." Again, the Examiner cites Fig. 4 in support. However, Fig. does not support the Examiner's contentions because as admitted in the final office action, Reiter reads and writes from the MDB tree is page-size pieces and even if one were to accept the Examiner's premises regarding segments and blocks (which are emphatically denied herein) the specific steps required in claim 14 cannot be done in page-size increments.

The Examiner further failed to show how Reiter teaches or suggests the claim step of creating an object description for an object by saving values owned by the object of the variables belonging to its class into said allocated block. As established above, Reiter neither teaches nor suggests object descriptions. The Examiner cites Fig. 5 in support and contends that the values of variables are items 118, 120, 122, 124, 126, 138, 144, and 146. However, the description of Fig. 5 identifies the above items as nodes. The Examiner contends that the first list corresponds to the key value table that in Fig. 5 point at subnodes. However the first list points at segments. Therefore, even using the Examiner's own presumptions, the key value table cannot be a first list. Moreover, it appears that Reiter does not teach or suggest keeping object descriptions apart form the objects as claim 14 requires and the Examiner has not shown otherwise.

The Examiner further failed to show how Reiter teaches or suggests the claim step of adding a new element to said second list containing the particular reference to said created object description. The Examiner cites fig. 8A in support. However, Fig. 8A does not show any such step. Step 160 relates to selecting a page into a page as part of a general method for maintaining the tree. Recall that the elements of the second tree are references to segments. The Examiner cites step in Fig. 8A deals with adding a new item of data and not a reference to a reference to a segment.

The Examiner further failed to show how Reiter teaches or suggests the claim step of determining the address of the object description of another object referenced in said object. The Examiner cites Fig. 5 in support and argues that the use of pointers implies that the object and the object description being referenced

have to be determined. See *In re Robertson*, 169 F.3d 743 (Fed. Cir. 1999). Anticipation cannot be established by implication without an explanation of how such implication occurs.

The Examiner further failed to show how Reiter teaches or suggests the claim step of setting the address of said respective object description as the reference in the created object description. The Examiner cites Fig. 8A and argued that this teaches storing a new pointer to a new page/block. Fig. 8B is discussed at col. 10, lines 1034 and there is no discussion whatsoever of setting the address of a new object description (there is none in Reiter) as the reference created in the created object description (again Reiter does not create an object description from an object created from information in volatile storage).

The Examiner further failed to show how Reiter teaches or suggests the claim step of storing said second list on said persistent storage. The Examiner draws the conclusion that the structure in Fig. 4 is written to persistent storage in pages. However, that overlooks the specifics of claim 14. That data are written to non-volatile storage as briefly discussed in Reiter does not mean that this happens as set forth in claim 14.

The Examiner further failed to show how Reiter teaches or suggests the claim step of storing the segments referenced by said first list on said persistent storage. The Examiner cited Fig. 4 and again cites the discussion where it is established that reiter reads and writes in page-long units. Using the Examiner's contention that the second list is a subnode table, then the second is not a page-long unit because figure 5 shows that pages include one or more nodes and a subnode table is only one part of even the smallest page.

The Examiner further failed to show how Reiter teaches or suggests the claim step of storing said first list on said persistent storage. Fig. 5 also proves that Reiter neither teaches nor suggests a step of storing a key value table because that table is not a page-long unit.

Reiter appears to concern a method for storing objects that determines whether there is sufficient space for storage of the object and then follows one of two paths depending on whether there is enough space.

The claimed invention does not concern the space availability issue; rather it specifies method steps once a determination has been made to store an object from volatile memory into persistent storage. The invention does not, as in Reiter, store page-long units of data. Rather, as stated in the specification at page 9, before an object can be persistently stored, a few steps are required to provide a respective infrastructure. The invention provides improved access to the stored objects by storing in persistent memory the first and second lists and by creating an object description that includes an address stored in persistent storage. Those steps are not found in the Reiter system. Specifically, the page 88 shown in figure 4 of Reiter includes neither a first or second list (segment map or object map) in the context of the claimed methods. The key values of Reiter are used to index tree nodes, not segments and the sub-node tables are used to index sub-nodes in the tree. See Abstract. Moreover, Reiter does not store differences between old addresses and new addresses for objects stored in persistent storage.

Reiter also fails to teach or suggest adding a new element as claimed. The Examiner contends that figure 8A of Reiter relates to this claim element but that flow chart relates to a method for determining whether a unit of data fits in a node of the

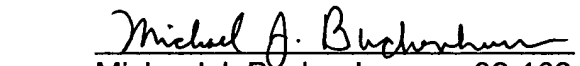
tree structure of Reiter. That is not the same as the claimed element of adding a new element.

Claims 15-23 are either directly or indirectly dependent on claim 14 and are patentable for at least the same reasons that claim 14 is patentable. Claim 24, as amended, is a computer program product counterpart of Claim 14 and is not anticipated by Reiter for the foregoing reasons.

CONCLUSION

For the foregoing reasons, Applicant respectfully requests reversal of the rejections and allowance of the pending claims and that a timely Notice of Allowance be issued in this case.

Respectfully submitted,



Michael J. Buchenhorner, 33,162
Holland & Knight LLP
Suite 3000
701 Brickell Avenue
Miami, Florida 33131
Telephone: 305/789-7773
Facsimile: 305/789-7799
Email: michael.buchenhorner@hklaw.com

CLAIMS APPENDIX

1. A method for restoring persistently stored objects of an object-oriented environment established in a computer system having a volatile memory and a persistent storage, the method comprising the steps of:

retrieving from said persistent storage a first list comprising first references to segments, stored in said persistent storage;

retrieving all segments referenced by said first references and storing them in said volatile memory;

saving in said first list the difference between an old memory address, at which the segment used to reside in the volatile memory, and a new memory address at which said segment is stored;

retrieving from said persistent storage a second list comprising second references to blocks, whereby said blocks contain an object description;

determining which segment contains said block referenced by a particular element of said second list;

creating an object in said volatile memory using said object description from said segment and saving a new address of said created object in said second list in volatile memory;

initializing said new object with values taken from said object description; and

determining said new addresses of said new object referenced by the newly-created object and setting said new address as the reference in said new object.

2. The method according to claim 1, whereby said first list and/or said second list are ordered lists.

3. The method according to claim 1 or 2, whereby said first list and/or said second list are organized as a B-tree.

4. The method according to claim 1, whereby the elements of said first ordered list are indexed by said first references.

5. The method according to Claim 1 one of the preceding claims, whereby each of said first references corresponds to the old memory address at which the respective segment used to reside in the volatile memory.

6. The method according to Claim 1, whereby the elements of said second ordered list are indexed by said second references.

7. The method according to one Claim 1, whereby each of said second references corresponds to the old memory address at which the respective block used to reside in said volatile memory.

8. The method according to Claim 1, whereby said object description is formed by a collection of values owned by an object for the variables belonging to its class.

9. The method according to Claim 1, whereby for each value in said object description of variables having a variable length the method further comprising the steps of:

allocating a number of blocks that allows to keep the actual value of the variable having variable length;

creating a linked list of said number of blocks;

saving said value into said number of blocks; and

storing a reference to the head of the linked list in said object description.

10. The method according to claim 1, whereby determining the segment that contains said block referenced by a particular element of said second list comprises the step of searching in said first ordered list (segment map) for the segment that contains said portion of said segment (block) referenced by said element.

11. The method according to claim 1, whereby determining the segment that contains said block referenced by a particular element of said second list further comprises the step of calculating the new address by adding the reference to said block (that corresponds to the old memory address) and said difference between said old memory address and said new memory address.

12. The method according to claim 1, whereby determining the new addresses of objects referenced by the newly created object comprises the step of searching in said second list (object map) for the element said contains the new

address of the referenced object, that is referenced by the old address of the respective object description.

13. The method according to claim 1, whereby for all references to heads of linked lists the method further comprising the steps of:

reading all blocks of said linked list;

allocating memory to store the value of the variable retrieved from the linked list; and

storing the value in said allocated memory.

14. A method for persistently storing objects of an object- oriented environment established on a computer system having a volatile memory and a persistent storage, the method comprising steps of:

allocating in said volatile memory, segments;

creating a first list comprising first references to said segments;

creating a second list comprising second references to blocks, wherein the blocks are portions of said segments;

allocating a block of one of said segments,

creating an object description for an object by saving values owned by the object of the variables belonging to its class into said allocated block;

adding a new element to said second list containing the particular reference to said object description;

determining the address of another object description of another object referenced in said object;

setting the address of said respective object description as the reference in the created object description;

storing said second list on said persistent storage;

storing the segments referenced by said first list on said persistent storage; and

storing said first list on said persistent storage.

15. The method according to claim 14, whereby said first list and/or said second list are ordered lists.

16. The method according to claim 14, whereby said first list and/or said second list are organized as a B-tree.

17. The method according to claim 14, whereby the elements of said first ordered list are indexed by said first references.

18. The method according to claim 14, whereby each of said first references corresponds to the current memory address at which the respective segment resides in the volatile memory.

19. The method according to claim 14, whereby the elements of said second ordered list are indexed by said second references.

20. The method according to claim 14, whereby each of said second references corresponds to the current memory address at which the respective block resides in said volatile memory.

21. The method according to claim 14, whereby determining the address of the object description of another object referenced in said object comprises the step of searching in said second ordered list for the element said contains the address of the object description of the referenced other object.

22. The method according to claim 14, whereby determining the address of the object description of another object referenced in said object comprises a step of using a reference to the respective object description provided by each object.

23. The method according to claim 14, whereby for each value of variables comprises a variable length the method further comprises steps of:

allocating a number of portions of one of said pieces of memory that allows to keep the actual value of the variable length variable;

creating a linked list of said number of portions;

saving value into said number of portions; and

storing a reference to the head of the linked list in said object description.

24. A computer program product stored on a computer usable medium, comprising computer readable program instructions for:

allocating in said volatile memory segments;

creating a first list comprising first references to said segments;

creating a second list comprising second references to blocks;

allocating a block of one of said segments,

creating an object description by saving values owned by the object of the variables belonging to its class into said allocated block;

adding a new element to said second list containing the particular reference to said created object description;

determining the address of the object description of another object referenced in said object;

setting the address of said respective object description as the reference in the created object description;

storing said second list on said persistent storage;

storing the segments referenced by said first list on said persistent storage; and

storing said first list on said persistent storage.

EVIDENCE APPENDIX

Upon information and belief, there is no evidence related to this case.

RELATED PROCEEDINGS APPENDIX

Upon information and belief, there no proceedings related to this case.

3345071_v2